Ontologies, Contexts, and Mediation: Representing and Reasoning about Semantics Conflicts in Heterogeneous and Autonomous Systems

Cheng Hian Goh, Stuart E. Madnick, Michael D. Siegel

Sloan WP# 3848 CISL WP# 95-04
August 1995

The Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

Ontologies, Contexts, and Mediation: Representing and Reasoning about Semantics Conflicts in Heterogeneous and Autonomous Systems*

Cheng Hian Goh† Stuart E. Madnick Michael D. Siegel
Sloan School of Management
Massachusetts Institute of Technology
Email: {chgoh,smadnick,msiegel}@mit.edu

Abstract

The Context Interchange strategy has been proposed as an approach for achieving interoperability among heterogeneous and autonomous data sources and receivers [25]. We have suggested [10] that this strategy has many advantages over traditional loose- and tight-coupling approaches. In this paper, we present an underlying theory describing how those features can be realized by showing (1) how domain and context specific knowledge can be represented and organized for maximal sharing; and (2) how these bodies of knowledge can be used to facilitate the detection and resolution of semantic conflicts between different systems. Within this framework, *ontologies* exist as conceptualizations of particular domains and *contexts* as "idiosyncratic" constraints on these shared conceptualizations. In adopting a clausal representation for ontologies and contexts, we show that these have an elegant logical interpretation which provides a unifying framework for *context mediation*: i.e., the detection and resolution of semantic conflicts. The practicality of this approach is exemplified through a description of a prototype implementation of a context interchange system which takes advantage of an existing information infrastructure (the World Wide Web) for achieving integration among multiple autonomous data sources.

Keywords: information infrastructure, interoperable federated systems, knowledge and data modeling, mediation, semantic interoperability.

*This work is supported in part by ARPA and USAF/Rome Laboratory under contract F30602-93-C-0160, the International Financial Services Research Center (IFSRC), and the PROductivity From Information Technology (PROFIT) project at MIT. Further information, including electronic versions of papers originating from the Context Interchange Project, is available on the WWW at `http://rombutan.mit.edu/context.html`.

†Funding from the National University of Singapore is gratefully acknowledged.

1 Introduction

Almost every statement we make is imprecise and hence is meaningful only if understood with reference to an underlying context which embodies a number of hidden assumptions. This anomaly is amplified in databases due to the gross simplifications we make (for efficiency reasons) in creating a database schema. For example, a database may record the fact

```
salary('Jones', 2000)
```

without ever explaining what "2000" means (what currency and scale-factor is used?), what is the periodicity (is this the daily, weekly, or monthly wage?), or what constitutes the person's salary (does it include year-end bonuses? what about overtime pay?). The real problem occurs when data sources and data receivers[1] maintain different assumptions about the data which are being exchanged: a data receiver formulates a query and interprets the answers returned in a certain context, whereas the query is executed by a data source which most likely provides the answers in a completely different context.

In a series of papers [10,23-25], we have described the *context interchange* approach as a viable strategy for achieving interoperability among heterogeneous and autonomous data systems. Siegel and Madnick [24,25] described the main ideas underlying context interchange with reference to a single-source single-receiver system where source and receiver differ in their assumptions on how data is represented or how it should be interpreted. Sciore et al. [22] described a superset of SQL, called *Context SQL*, which allows one to define data contexts as part of a query and to formulate queries predicated on data contexts. In a further extension [23], they have proposed the notion of *semantic values* as a basis for data exchange and discussed a number of related issues, such as properties of conversion functions which are needed to realize transformations on semantic values. Goh et al. [10] compared the context interchange strategy with other dominant integration strategies described in the literature and highlighted advantages this approach has over the others. An earlier attempt at transitioning these ideas to an actual implementation in a client/server environment has been reported in [8].

In this paper, we are primarily concerned with how semantic knowledge can be represented (in the form of contexts and ontologies) and what inferences can be made with them in facilitating the mediation of semantic conflicts. To provide a concrete basis for our discussion, we begin by describing the architecture and implementation of a Context Interchange Prototype. This implementation capitalizes on the infrastructure of the World Wide Web and is aimed at providing

[1] Data sources refers to databases, data feeds, and other applications which provide structured data upon request; data receivers refer to users, consolidating databases (e.g., data warehouses), and applications that make these requests.

mediated access to a wide variety of data sources. The inherent heterogeneity of these data sources provide many rich and interesting examples and constitutes a major driving force in developing the theory.

Following a description of the prototype, we present a framework which examines the differences between alternative integration paradigms from the perspective of the user issuing the query. Virtually all of the integration strategies proposed in the literature fall in one of the two categories: systems can be *tightly-coupled* or *loosely-coupled.* In the case of tightly-coupled systems, a shared schema of some form is used to encapsulate the underlying conflicts and hence all queries must be formulated using this shared schema (or some view of it) [1,2,6,14]. With loosely-coupled systems, there is no shared schemas; instead the user must take on the responsibility for detecting and reconciling potential conflicts [15]. We will show that the context interchange strategy does not induce such a dichotomy: instead, it accommodates both approaches while achieving semantic interoperability using the services provided by a Context Mediator.

Section 4 presents the representational issues concerning contexts and ontologies. An object-centric representation with a simple logical interpretation is introduced. *Ontologies* are coherent models of specific domains of discourse; *contexts* on the other hand describe what is true of the specific environments of both sources and receivers.

Section 5 introduces the abduction framework which is used as the basis for conflict detection. We present an algorithm which compares two contexts and outputs a *conflict table.* This conflict table forms the basis for transforming the original query to an equivalent one which circumvents the semantic conflicts arising from the data exchange.

Finally, the last section summarizes our contribution and outlines some of the outstanding research issues.

2 A Web-Based Context Interchange Architecture

In a short period of time, the World Wide Web has experienced phenomenal growth and is poised to be the de facto global information infrastructure. Information of every conceivable kind (including structured data originating from legacy data systems) is being made available on the Web with more being added each day. These changes bring about a great challenge for the "database integration" community: we are swarmed with data originating from different contexts having disparate interpretations, all begging to be understood.

The goal of the Context Interchange Prototype is to address the challenge of providing semantic interoperability between sources and receivers of "structured" data on the World Wide Web. We have in mind the following criteria for success: (1) such a system (strategy) should be

scalable; i.e., there should not be significant degradation of functionality despite the large number of data sources and receivers involved; (2) the system should be capable of evolving "gracefully" in response to changes while remaining easy to administer; and (3) where possible, it should take advantage of existing services in the underlying information infrastructure.

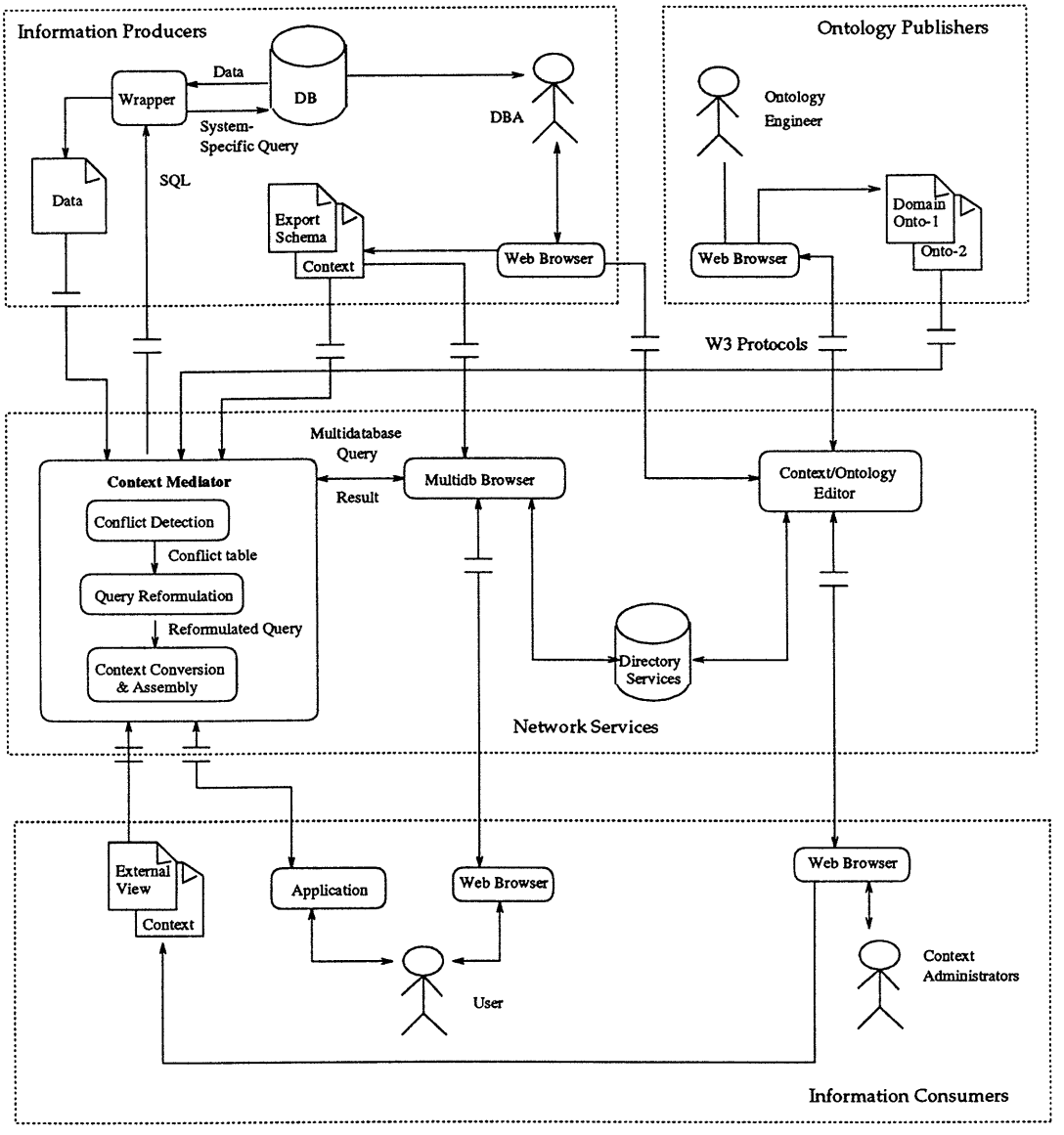

Figure 1: A Web-Based Architecture for Context Interchange.

Figure 1 shows the architecture of the Prototype which is currently being implemented[2]. To

[2]A beta release of the system encompassing most of the functionalities described here will be made available in Fall 1995. At the time of writing, this prototype can be reached via the URL

allow it to scale, this system is designed without any centralized facility. Anyone who has access to the World Wide Web can publish information, and anyone (who has the proper authorization) can read it. To publish data, a source merely needs to define what data elements are available in an *export schema*, and installs a *wrapper* which provides a well-defined interface between the underlying source and the external world[3]. To gain access to data, a receiver needs to execute a client program. Client programs can query data sources directly by sending a (SQL) query to the respective wrappers. These wrappers translate the SQL to an equivalent query which is executed by the underlying system, and returns the answer in a tabular form. In this manner, wrappers help insulate users from heterogeneous system interfaces by allowing all queries to be issued using a standard query language (SQL) based on a canonical data model (the relational data model). For example, we have implemented wrappers for various Web-based information services such as the Networth quote server[4]. Instead of physically interacting with the on-screen form provided by the Networth quote server, one can issue an SQL query to our wrapper which translates the incoming query to a parameterized URL understood by the quote server. The wrapper also captures the (text) document returned by the quote server, "scrapes" the data requested (i.e., attributes which appeared in the target list of the original query), and return these in a tabular form. For instance, to get the latest trade price of IBM's stock, one could enter the following URL:

```
http://context.mit.edu/cgi-bin/networth.cgi?query=select+
networth.companyName,+networth.LastTrade+from+networth+where+
networth.Ticker+%3D+IBM[5]
```

Similar wrappers for relational (Oracle) databases also have been implemented. These serve as gateways to relational databases and provide the means of sending a query to the underlying DBMS using the HTTP-protocol. In this case, however, there is no need to perform query translation since the SQL query can simply be passed along to the underlying DBMSs.

While constituting a powerful underlying mechanism, parameterized URLs are cumbersome and should not be the interface that users have to deal with. To aid in the actual formulation of

---

`http://context.mit.edu/cgi-bin/mint.cgi`. You are welcome to try out some of these functions. However, bearing in mind that this is our development environment, it may not be very stable. If you encounter problems, feel free to email the authors and we will be happy to attend to them.

[3] In actual fact, the wrapper can be provided by someone other than the source. For instance, our prototype implementation provide access to the Networth quote server via a wrapper which "runs" on our web-site (more about this latter). The current implementation also include a *generic wrapper* which can be configured to "wrap around" different data sources by changing the accompanying configuration file.

[4] The Networth quote server can be reached at `http://quotes.galt.com/`. It provides (amongst other things) stock quotes on stocks traded in major US stock exchanges on a real time basis (with a 15 minutes delay).

queries, we have also implemented a multidatabase browser which provides a QBE-like interface for constructing an SQL query. The multidatabase browser maintains a directory of all data sources registered with it and allows users to both select the data sources they like to access and incrementally (interactively) build up an SQL query by adding constraints. When the user is satisfied with the query, the multidatabase browser hands it to a query processor. If the query involves only a single source, it will be shipped directly to the wrapper for that source; otherwise, it is handed to a query planner which transforms the original query to a series of subqueries which can be executed at the sources. The results from the sources are then assembled before they are sent to the multidatabase browser.

While interesting by itself, the above scheme merely provides access to data in the form which is present in different contexts. Our real objective, however, is to provide data from disparate sources without constant intervention from users in resolving the underlying semantic conflicts.

There are two pre-requisites for mediation of semantic conflicts: (1) all sources and receivers must describe their contexts explicitly with respect to a collection of shared ontologies; and (2) all queries must be routed through a *Context Mediator*. The creation of contexts and ontologies is aided by a Context/Ontology Editor[6] which provides hypertext traversal of concepts and links. With proper authorization, these documents can be altered using a form-based frontend. It is worth noting that all of the information (export schemas, ontologies, contexts) may be colocated with the data source (where authorized). A Directory Service identifies what resources (data sources and ontologies) and data receivers (users and applications) are registered with the system and supply appropriate pointers (URLs) to all other needed information. In this manner, DBAs and ontology publishers retain complete control over the information they provide. Multiple mediators can exist in the system providing greater resilience to failure and achieving better throughput.

A Context Mediator does a number of things each time it receives a query referencing multiple data sources. First, it compares the contexts of the data sources and receivers to determine if semantic conflicts exist, and if so, what conversions need to take place to resolve them. This is referred to as *conflict detection* and the information detailing the conflicts are presented in a *conflict table*. Second, the original query is transformed to an equivalent one in which all semantic conflicts are reconciled by inserting the appropriate conversion operations identified in the conflict table. This query then undergoes an optimization process. Optimization takes a number of forms: a subquery can be reformulated so that constants referred to in a query are

[6]This editor is inspired by the Ontology Editor at Stanford KSL:
`http://www-ksl-svc.stanford.edu:5915/`. Unfortunately, we find it difficult to directly reuse their tool given the differences in our conceptualization of ontologies and contexts. More details of these differences can be found in Section 4.

Worldscope:AF

company_name	ticker	location_of_incorp	net_income	total_assets
DAIMLER-BENZ AG	DAI	GERMANY	346577	56559478
INTERNATIONAL BUSINESS MACHINE	IBM	NEW YORK	-8100000	81113000
NIPPON TELEGRAPH & TELEPHONE C	NTT	JAPAN	486515	119190708
⋮	⋮	⋮	⋮	⋮

Networth:stock

companyName	ticker	lastTrade
Daimler-Benz Atkieng ADS	DAI	49
Intl Bus. Machines	IBM	105 7/8
Nippon Tel & Tel ADS	NTT	45 1/2
⋮	⋮	⋮

Almanac:US_States

state	abbrev
CALIFORNIA	CA
MASSACHUSETTS	MA
NEW YORK	NY
⋮	⋮

Almanac:Currencies

Country	Currency
GERMANY	Deustche Mark
JAPAN	Japanese Yen
UNITED STATES	U.S. Dollar
⋮	⋮

Almanac:Exchange

source	target	rate
Deustche Mark	British Pound	0.443547
Japanese Yen	British Pound	0.006829
U.S. Dollar	British Pound	0.624999
⋮	⋮	⋮

Figure 2: Fragments of sample tables supporting the example.

converted to the context of the data source executing that query, subqueries can be resequenced and/or data conversions can be merged in a way that seeks to minimize the costs of conversions. Finally, the intermediate answers obtained from component systems must be merged together and converted to the context expected by the receiver.

3 Querying Heterogeneous Databases

To better elucidate the features offered by the Context Interchange strategy, we will present an example highlighting the various options a user faces in querying heterogeneous data sources.

Figure 2 shows fragments of some of the actual data sources that are accessible from the Context Interchange Prototype and are available to a user. Suppose the user is interested in the answers to the query: "what are the 'net income' and 'latest trade price' for companies whose 'total assets' are greater than 1 billion?". There are several different ways by which this query may be asked, depending on what services the "system" provides. These possibilities are shown in Figure 3.

Approach 1: *No* Shared Schema, *No* Context Mediation. In a multidatabase language

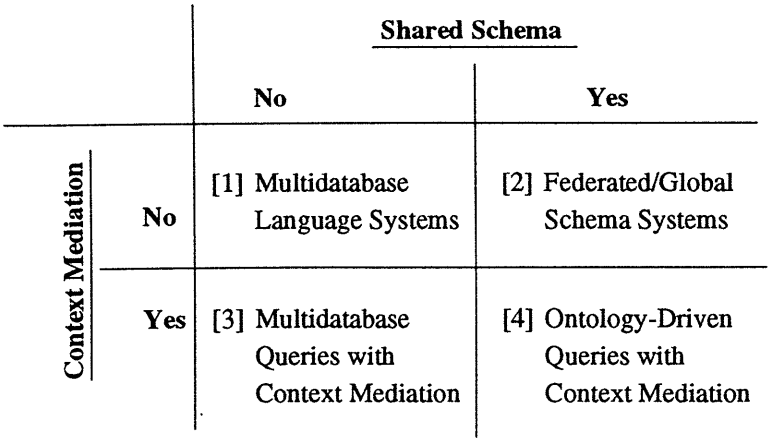

Figure 3: Different ways of asking the same query.

system [15], the user formulates a query using the export schemas of the data sources and takes on the responsibility for detecting potential conflicts as well as determining how those conflicts can be circumvented. Within such an environment, a query to accomplish this might be:

```
select w.company_name, $income, n.lastTrade
from Worldscope:AF w, Networth:stock n, Almanac:exchange e,
    Almanac:currencies c
where w.total_assets > $one_billion and w.location_of_incorp = c.country
and e.source = c.currency and e.target = 'British Pound'
and $one_billion = 1000000 / e.rate
and $income = w.net_income * e.rate * 1000
and w.ticker = n.ticker
union
select w.company_name, $income, n.lastTrade
from Worldscope:AF w, Networth:stock n, Almanac:exchange e,
    Almanac:US_States s
where w.total_assets > $one_billion and w.location_of_incorp = s.state
and e.source = 'U.S. Dollar' and e.target = 'British Pound'
and $one_billion = 1000000 / e.rate
and $income = w.net_income * e.rate * 1000
and w.ticker = n.ticker
```

This contorted expression is the result of anomalies in the underlying sources as well as certain assumptions held by the user in interpreting the data. Specifically:

1. The actual currencies used for reporting `total_assets` and `net_income` is not found in `Worldscope:AF` and must be obtained from our `Almanac` databases. This is accomplished by doing a join between `location_of_incorp` and `Almanac:currencies.country` to find out what is the currency used in the particular country.
2. Even then, the attribute `location_of_incorp` reports the state in which the company is incorporated as opposed to just "United States" when the companies under consideration are incorporated in the U.S. This forces the user to write the query in two parts: one for foreign companies and one for U.S. companies.
3. The user is located in England and assumes a currency of British pounds.
4. The scale-factor used in Worldscope is 1000's; however the user assumes `net_income` and `tradePrice` to be in 1's; and `total_assets` to be in 1000's.
5. Company names are reported differently in Worldscope and Networth and hence a join between the two databases using company name will not yield the expected result. It turns out that both report an overlapping set of ticker symbols (for companies traded in U.S. exchanges). Hence, the ticker symbol is used in place of company names for doing the join.

Notice also that for efficiency purpose, the constant "1 billion" is converted to the currency and scale-factor corresponding to each tuple so that the selection can be done first. The alternative, which is to retrieve and convert the entire database to the user context (i.e., British Pounds with a scale-factor of '1') before performing the selection, is likely to be prohibitively expensive.

It should be apparent from the preceding example that while this approach may provide greater flexibility and control to the user, it can be unrealistic in practice since it requires users to have full knowledge of the semantic disparities and idiosyncrasies in underlying sources and ways to resolve these problems.

Approach 2: *With* Shared Schema, *No* Context Mediation. The alternative paradigm, which has been widely reported in the literature, is to have some specialist (perhaps the DBA) reconcile all conflicts *a priori* using a shared schema. In the above example, this might be accomplished by first defining the shared schema `myView`:

create view `myView` **as**

```
select w.company_name, $net_income, $total_assets, n.lastTrade
from Worldscope:AF w, Networth:stock n, Almanac:exchange e,
    Almanac:currencies c
where w.location_of_incorp = c.country
and e.source = c.currency and e.target = 'British Pound'
and $net_income = w.net_income * e.rate * 1000
and $total_assets = w.total_assets * e.rate
and w.ticker = n.ticker
union
select w.company_name, $income, $total_assets, n.lastTrade
from Worldscope:AF w, Networth:stock n, Almanac:exchange e,
    Almanac:US_States s
where w.location_of_incorp = s.state
and e.source = `U.S. Dollar' and e.target = 'British Pound'
and $net_income = w.net_income * e.rate * 1000
and $total_assets = w.total_assets * e.rate
and w.ticker = n.ticker
```

The user query may then take the form:

```
select company_name, net_income, lastTrade from myView
where total_assets > 1000000
```

Examples such as the one above have often been used as illustration for the power of a shared schema. A number of prototype systems employing this strategy have also been reported in the literature, differing largely in the choice of the underlying data model (e.g., relational in the case of Multibase [14], an object-oriented data model in the case of Pegasus [1], and knowledge representation languages in the case of Carnot [6] and SIMS [2]).

It is interesting to note that the literature has largely been silent on how a query issued against such a view can be reformulated to execute efficiently. In traditional databases, this is probably less of an issue given that the underlying data are relatively homogeneous. This however is not true of heterogeneous database systems.

From a users' perspective, having all queries mediated by a shared schema is not always desirable. In some instances (e.g., the financial industry), it is imperative that users have explicit knowledge of what sources are used for specific pieces of data since this knowledge has great bearings on one's confidence in the accuracy and reliability of the information [27]. In many instances, the

use of a shared schema causes this information to be obscured or unavailable. Allowing users direct access to export schemas is also useful when not all queries can be anticipated. In many instances, users often have greatly diversified data requirements; the creation of a shared schema to encapsulate everyone's needs may be impractical. This is particularly true for queries aimed at exploiting *features* of data sources which are treated as *bugs* to be eradicated in the shared schema. (Examples of this will be provided at the end of this section.) Based on the above concerns, the Context Interchange strategy aspires to provide mediation services independent of whether or not queries are formulated against an underlying shared schema[7].

Approach 3: *No* Shared Schema, *With* Context Mediation. Under this approach, users have the flexibility of formulating their queries directly using the export schemas for the underlying sources without the need to explicitly identify or mediate the conflicts. With reference to the above example, a user, having the appropriate context defined (see Section 4.2), may issue the following query:

```
select w.company_name, w.net_income, n.lastTrade
from Worldscope:AF w, Networth:stock n
where w.company_name = n.companyName and w.total_assets > 1000000
```

In this case, the user can remain completely ignorant of the different currencies and scale-factors used for reporting `net_income` and `total_assets`, as well as the different naming conventions used for company names. When the query is submitted to a Context Mediator, the latter examines the contexts associated with the data sources and the receiver, identifies these conflicts, and transforms the user query to an equivalent query which returns the required answer.

Approach 4: With Shared Schema *and* Context Mediation. The equivalent of a shared schema in the Context Interchange approach is the set of shared ontologies; this approach was adopted for the first prototype implementation and is described in [8]. Our use of ontologies in this sense is somewhat similar to the domain model in SIMS [2]. However, instead of creating one big domain model encompassing all of the relevant knowledge in the integration domain, we advocate the use of separate ontologies, each encompassing a well-defined domain. These can be created in a modular fashion allowing them to be combined in different configurations to support different integration activities. (This is consistent with the efforts of the Knowledge Sharing community [11,20].) For all intent and purposes

[7]There are other arguments, from a system administration point of view, for why a shared schema approach as traditionally implemented is not desirable. These problems will be elaborated later in Section 4.2.

of a user, the underlying domain model(s) can function as a shared schema against which queries can be formulated. Such a query will look no different from the simple shared-schema query (of Approach 2) shown above. (There are however important differences in how conflicting semantics are identified and reconciled. These are the subject matter of the next two sections.)

All of the four approaches for formulating a query can co-exist in a Context Interchange system and will be supported in the current prototype implementation: an application can bypass available mediation services altogether, or use a context mediator in connection with a query on either a set of export schemas or some import (shared) schema. For example, the multidatabase browser provides access to both export schemas and import schemas (views defined on the ontology or a collection of export schemas); any combination of these can be used as the basis for constructing a query. In the current implementation, a designated "button" on the browser's screen allows users to turn on or off context mediation. In addition, a user can optionally select a local context to which results will be converted. It is in fact possible for a user to turn on mediation without a local context: in this case, data exchange between sources will be mediated, but the answer will be returned in its original context (e.g., if `net_income` from Worldscope is returned, it will be in the currencies as reported in Worldscope using a scale-factor of 1000; meanwhile, the "join" on company names between Worldscope and Networth will be mediated).

As an aside, it should be pointed out that users can have control over what is being mediated in a query. This is often useful in queries aimed at exploiting features of the underlying system. For example, one may be interested to know which companies have names that are represented in the same way in both Worldscope and Networth. The query

```
select w.company_name from Worldscope:AF w, Networth:stock n
where w.company_name = n.companyName
```

achieves this only if mediation is turned off when company names are being compared. In other instances, finer granularity of control may even be desired. For example, pre-tax net income may be reported in one source and post-tax net income in another. Under normal circumstances, the Context Mediator will seek to mediate this conflict when net income data is being exchanged between the two sources. A user may want to exploit the features of the two sources to find the companies which are tax-exempted. This can be accomplished with the query:

```
select R.company_name from DB1:R, DB2:S
where R.net_income =[!tax-status] S.net_income
```

In this instance, net_income is mediated for their differences (e.g., currencies, scale-factors) except tax-status. Some of these extensions (and others) to SQL have been reported elsewhere [22,23].

4 Representing Semantic Knowledge for Context Mediation

The premise that data elements that are exchanged across contexts are values drawn from *semantic domains* is key to Context Interchange. The theory of *semantic values* has been described in [23]. A semantic value is a "compound" object having an interpretation which is modified by values of tags (*modifiers*) that are associated with it. For example, the two semantic values

1(unit='meter')

and

100(unit='centimeter')

are obviously equivalent. However, this equivalence is clear only because the assumption concerning the underlying unit of measurement is made explicit. In many legacy systems (whether they are data sources or receivers), the assumptions concerning how data should be interpreted are often inaccessible or deeply buried within some application code. The strategy adopted in the Context Interchange approach is to capture these assumptions explicitly through the use of *semantic domains*, from which semantic values are drawn from.

The viability of the Context Interchange approach for large-scale interoperability is largely contingent on how knowledge of data semantics can be represented effectively. The decisions underlying the organization and representation of semantic knowledge in a Context Interchange system have been influenced by a number of design criteria, most of which have been described elsewhere [10]. Two of these are worth mentioning here: (1) given that the process of eliciting hidden assumptions about data is an expensive one, this knowledge should be represented in a form that is sharable and reusable; and (2) knowledge of data semantics should be represented in a form in which relevant inferences can be made efficiently. These concerns led us to a three-tier organization which include ontologies, contexts, and schematic mappings.

4.1 Ontologies

Within the artificial intelligence (particularly, knowledge representation) literature, *ontologies* refer to formal conceptualizations of specific domains. Recent interests in "ontological engineering" has led to attempts at creating a library of sharable ontologies (e.g., engineering mathematics, semiconductor products, and vertical transportation) [11,12].

While much of the work from this community has relevance to us, the ontologies created for Context Interchange differ in the attention we give to semantic values. Specifically, an ontology for Context Interchange consists of a collection of semantic domains and the "relationships" which exists among these. Figure 4 shows a graphical representation of a fragment of the Physical-Quantities ontology from [12] that is adapted for our purpose. Examples of semantic domains in this ontology are `physical_quantity`, `monetary_quantity`, and `monetary_unit`. Instances (i.e., semantic values) of the domains to which they belong are connected to the latter with a dotted line. For example, `'U.S. Dollars'` and `'Japanese Yen'` are both instances of the semantic domain `monetary_unit`. Semantic domains relate to one another via binary *links* (or relationships) which can be labeled: for instance, `unit` and `dimension` are both labeled links, with the first being a relationship between `physical_quantity` and `unit_of_measure`. Certain relationships can be designed as *modifiers*. For example, both `unit` and `scale_factor` are modifiers for `physical_quantity`: hence a value from this domain can have multiple interpretations, depending on the value assigned to the modifier. Thus, the semantic values `1(unit=kilometer,scale_factor=1)` and `1(unit=mile,scale_factor=1000)` are not the same even though both has a base value of `1`.

Semantic domains and even relationships can be organized in a taxonomy: a `monetary_quantity` is a type of `physical_quantity`, and a `currency` link is a type of `unit` relationship. The presence of a taxonomy means that "properties" of a given "object" can be inherited by those which are subclass to it. For example, since `unit` is a modifier for `physical_quantity` and `monetary_quantity` is a subclass of the latter, this means that `unit` is also a modifier for `monetary_quantity`. In this case though, the link `unit` has been given a different label (`currency`). Largely for the purpose of getting around the limitations imposed by binary-only relationships, we allow reified links, called *entity sets*, to exist in our ontologies. An example of an entity set in the "Physical Quantities" ontology is `currency_exchange`, which provides a way of describing the three-way relationship between `source`, `target`, and `exchange_rate`. In summary, our ontologies are organized as collection of classes (semantic domains), their instances (semantic values), and relationships (regular links, modifiers, taxonomic relationships, and entity sets).

The structure we have just described is amenable to a number of physical implementations (e.g., object-oriented data model, frame-based knowledge representation system). In the current implementation, we have chosen a Horn clause [16] representation which makes it easy for performing the kind of inferences which we will described in the next section. These clauses take the form

$$head \leftarrow body$$

where *head* is a predicate and *body* is a conjunction of predicates (possibly none). All free vari-

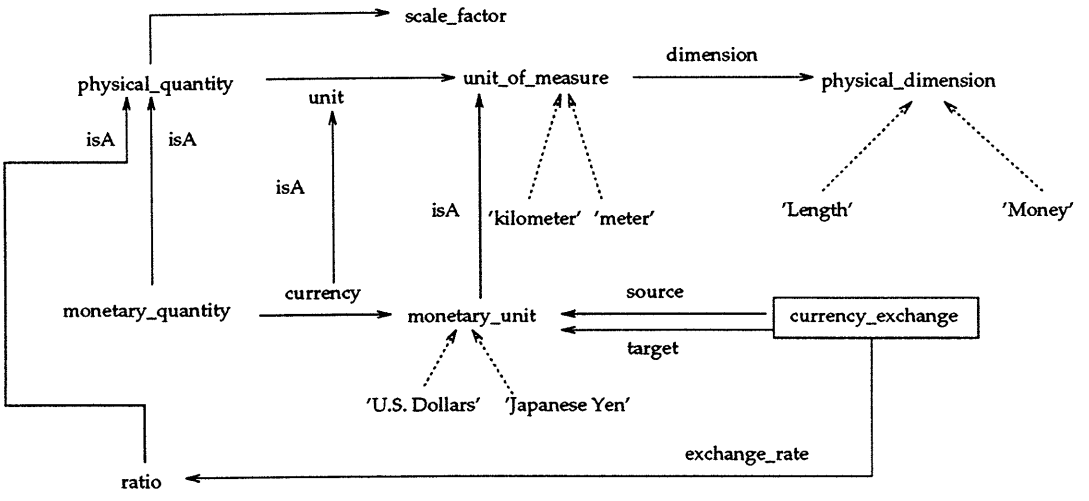

Figure 4: A conceptualization of the domain of measurements involving physical quantities. (Entity sets are "boxed" to distinguish them from semantic domains.)

ables in the head are assumed to be universally quantified; all others are existentially quantified. For example, the clause

$$\texttt{monetary\_unit}(X) \leftarrow \texttt{unit\_of\_measure}(X) \wedge \texttt{dimension}(X, \texttt{'money'})$$

states that "forall X, if X is an instance of the semantic domain `unit_of_measure` having `dimension 'money'`, then X is an instance of the semantic domain `monetary_unit`". The converse turns out to be true as well. Hence, we may also write in the ontology:

$$\texttt{dimension}(X, \texttt{'money'}) \leftarrow \texttt{monetary\_unit}(X)$$
$$\texttt{unit\_of\_measure}(X) \leftarrow \texttt{monetary\_unit}(X)$$

The clausal representation thus provides us with a parsimonious representation for both necessary and sufficiency conditions for the descriptions of semantic domains which are not unlike description languages such as Loom [18], Kif [20], or Classic [4][8]. The clauses corresponding to the ontology fragment shown earlier can be found in Figure 5.

As was also described in [23], conversion functions define equivalence between two distinct semantic values having different modifier values. For example, converting `1(unit='meter')` to `100(unit='centimeter')` requires multiplying the base value ('1') by '100'. Conversion

[8] Our Horn clause representation appears to be less expressive only because it does not support cardinality constraints on roles. For instance, it is extremely difficult to come up with a concise way for constraints like `(define-concept rich-kid :is (:and person (:at-least has-car 3)))` [3].

```
/* declaring the categories of things */
entity(currency_exchange).
semval(physical_quantity).
semval(monetary_quantity).
semval(unit_of_measure).
semval(monetary_unit).
semval(scale_factor)
semval(physical_dimension).
semval(ratio).
link(dimension,unit_of_measure,physical_dimension).
link(unit,physical_quantity,unit_of_measure).
link(currency,monetary_quantity,monetary_unit).
link(scale_factor,physical_quantity,scale_factor).
link(source,currency_exchange,monetary_unit).
link(target,currency_exchange,monetary_unit).
link(exchange_rate,currency_exchange,ratio).
modifier(unit).            /* modifier(currency) is implied */
modifier(scale_factor).
/* axioms */
physical_quantity(X) ← monetary_quantity(X).
physical_quantity(X) ← ratio(X).
unit(X,Y) ← currency(X,Y).
unit_of_measure(X) ← monetary_unit(X).
dimension(X,'money') ← monetary_unit(X).
monetary_unit(X) ← unit_of_measure(X) ∧ dimension(X,'money').
monetary_quantity(X) ← physical_quantity(X) ∧ unit(X,Y) ∧ monetary_unit(Y).
monetary_unit(Y) ← monetary_quantity(X), unit(X,Y).
/* instances */
monetary_unit('Japanese Yen').
monetary_unit('U.S. Dollar').
unit_of_measure('kilometer').
unit_of_measure('meter').
physical_dimension('length').
physical_dimension('money').
dimension('kilometer','length').
dimension('meter','length').
```

Figure 5: Clausal representation for the fragment of "Physical_Quantities" ontology shown in Figure 4.

functions are an integral part of the ontology: modifier links in an ontology may name one or more conversion functions, thus making it possible for conversion to take place. In the preceding example, we name the unit conversion function explicitly with the clause

`conversion_function(unit,cvtUnit).`

Conversion functions may be implemented in a variety of ways: these range from simple arithmetic manipulations (e.g., "multiply by 100" as when dealing with differences in scale-factor), to complex operations which can only be realized using external programs (e.g., "dvips" for converting between ".dvi" files and postscript files). Hence, `cvtUnit` can be implemented in the form of axioms associated with the ontology. For example, we may write the following axiom for realizing the conversion from `'kilometer'` to `'meter'`:

$$\texttt{cvtUnit}(X,\texttt{'kilometer'},\texttt{'meter'},X') \leftarrow X' = X * 1000$$

In the case of currency conversions, the conversion function can be defined with reference to the "Physical Quantities" ontology in the following manner:

$$\texttt{cvtCurrency}(X,Src,Tgt,X') \leftarrow \texttt{currency\_exchange}(E) \wedge \texttt{source}(E,Src) \wedge \texttt{target}(E,Tgt) \wedge \texttt{exchange\_rate}(E,R) \wedge X' = X \times R.$$

We have made the simplifying assumption here that fluctuations in exchange rates are not important in this domain. In many instances, conversion functions are seldom so straight-forward and may involve additional parameters (e.g., date and time). Also, there may be more than one way of converting from one representation to another. In the case of currency conversions, the exchange rate used for performing conversion may be a policy-dictated figure which is only revised on occasion (e.g., as applied for internal transfer pricing purposes), or it may be necessary to used the latest available figure right down to the last second (as in money markets). All of these variations can be captured as part of the underlying ontologies and contexts. Exactly which conversion function is used is largely determined by the needs of the users and can be specified as part of a user's context. The key observation here is that conversion functions are "pegged" to the underlying ontologies and contexts as opposed to specific data sources. The decision as to where this information (e.g., exchange rate) should be obtained can be made later (at runtime) since the cost of actually getting the data is likely to vary from one user to another. (The mapping from data sources to ontologies are enabled by *schematic mappings*, as described in Section 4.3.)

Figure 6 shows a (simplified) fragment of a "Financial Reporting" ontology used for the integration of databases accessed by the prototype implementation. Notice that in addition to serving as reified links, entity sets also provide an *aggregation* facility [26] by allowing several

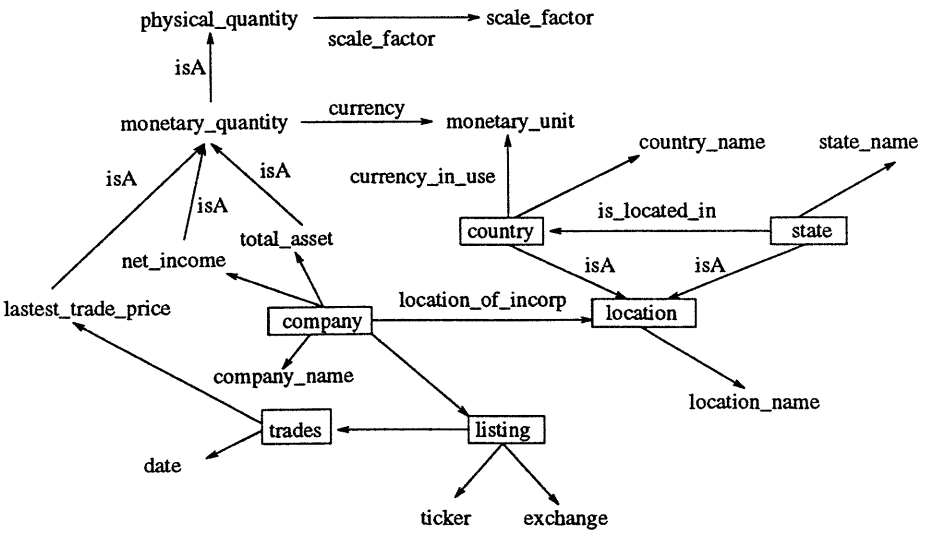

Figure 6: A conceptualization of financial reporting domain.

semantic domains to be collected together and asserted as attributes (properties) of some objects in the real world. For example, without this facility the notion of a "company" has no inherent representation in our ontology: we may know of its name, where it is located, what business it is in, but none of these things is a "company". Notice also that this ontology makes use of concepts which are defined in the "Physical Quantities" ontology. Similar to the model set forth in [12], we reject the adoption of a single ontology finessing all details of a complex world. Instead, we advocate that ontologies should be created for finite and well-delimited domains which can be assembled together for the problem at hand. In other words, an ontology can reference concepts and relationships defined in another through simple declaration of the intent. In the current prototype, each ontology is represented as a named "module", each of which is a collection of Horn clauses. This provides for a clean separation between different ontologies and allow them to be reused in different situations. An example of the relationships between different ontologies in the current implementation is illustrated in Figure 7.

4.2 Contexts

Up to this point, we have used the word "context" informally in referring to the particular circumstances which a data source or receiver finds itself in. For instance, we may say "in the context of this database, all monetary quantities are assumed to be in U.S. Dollars." In the Context Interchange approach, "contexts" also refer to collections of formal statements that are used in elucidating the meaning "things" have for particular environments.

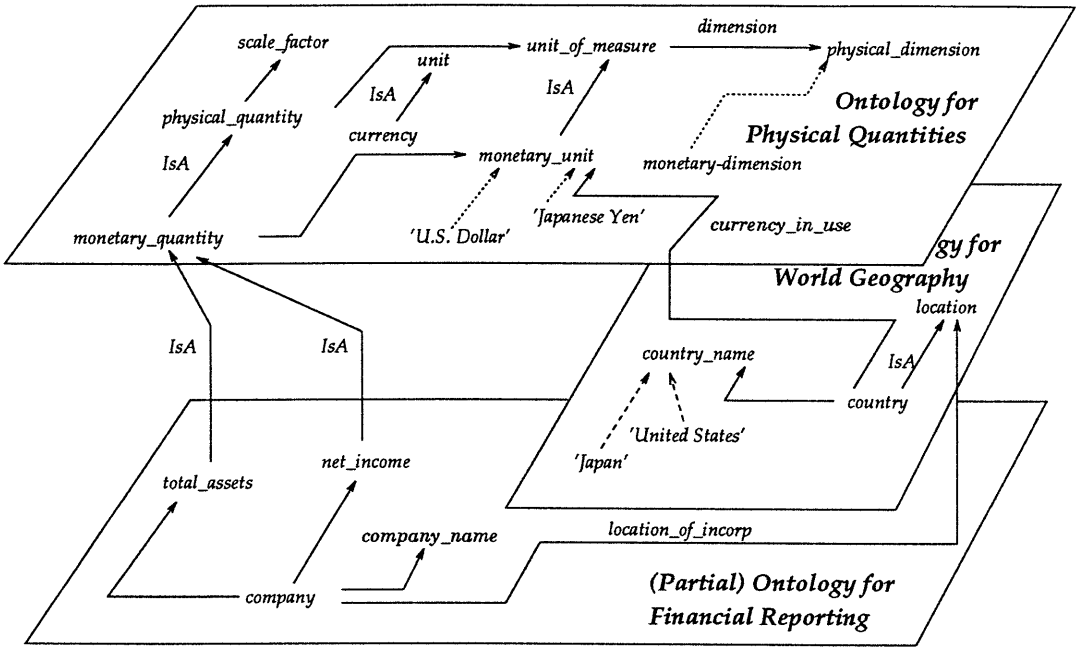

Figure 7: Graphical rendition of how ontologies can be combined.

In order for statements within a given context to be meaningfully compared to those in a different context, these sentences must be formulated using a common vocabulary. For example, we may make assertions that "dates" are reported using the format "DD/MM/YY" in one context, and "MM/DD/YY" in another. For these assertions to be meaningfully compared, both contexts must agree that "date" refers to a day in the Julian calendar as opposed to say, a meeting between a boy and a girl; furthermore, they must also mutually agree that "MM", "DD", and "YY" refer to month, day and year respectively. This agreement is achieved by having the two contexts make assertions with references to one or more shared ontologies which define these concepts (semantic domains) unambiguously.

Two categories of assertions can be found in a context: those which *retract* statements previously asserted (in some *outer*[9] context or within some ontology referenced by the context), and those which *add* to those statements.

A context may reference a shared ontology and yet disagree with some portion of the underlying conceptualization. For example, an ontology may make a distinction between pre-tax income and post-tax income in a financial statement. When this distinction is irrelevant to a local environment, these may be retracted in the local context. More commonly, a context may retract

[9]Contexts can be *nested*: context c_1 is said to be the *outer* context for context c_2 if c_2 is nested in c_1. This notion will be further elaborated on.

statements asserted in some other context which it inherits. For example, the outer context may suggest that all currency amounts are reported in some currency, but the local context might have reason to do otherwise.

Context: WORLDSCOPE

references('financial_reporting').

scale_factor(X,1000) $\leftarrow$ monetary_quantity(X).

company_financials(X, Y) $\leftarrow$ net_income(X, Y).

company_financials(X, Y) $\leftarrow$ total_assets(X, Y).

currency(X,'U.S. Dollar') $\leftarrow$ company_financials($Comp, X$) $\wedge$
location_of_incorp($Comp, State$) $\wedge$ is_located_in($State, Ctry$)
$\wedge$ country_name($Ctry$,'United States').

currency(X, Cur) $\leftarrow$ company_financials($Comp, X$)
$\wedge$ location_of_incorp($Comp, Ctry$)
$\wedge$ currency_in_use($Ctry, Cur$).

$\vdots$

Context: USER

references('financial_reporting').

scale_factor(X,1) $\leftarrow$ net_income(X).

scale_factor(X,1000) $\leftarrow$ total_assets(X).

scale_factor(X,1) $\leftarrow$ latest_trade_price(X).

currency(X,'British Pound') $\leftarrow$ monetary_quantity(X).

$\vdots$

Figure 8: Portions of selected contexts for the earlier example (defined with reference to the ontology shown in Figure 6.

In most situations, contexts serve to augment what is already known through ontologies and other outer contexts. A context may define additional "objects" based those already known. For most purposes, the assertions that are of most interest to us are those which provide assignments of modifier values to semantic domains. Regardless of what they do, these assertions take the same (Horn clause) representation as in ontologies. Figure 8 shows portions of two contexts pertaining to the example we presented in Section 3: the context WORLDSCOPE is associated with the Worldscope database, and the context USER is the one used by the user issuing the query.

As shown in Figure 8, the WORLDSCOPE context references an ontology called FINANCIAL_REPORTING and supplements it with additional axioms which take effect in this context. For instance, scale_factor for all monetary_quantitys is set to '1000'. This context also adds a new link, called company_financials, which is defined to encompass net_income and total_assets. This new link is used in facilitating the assignment of currencies: the first states that if location_

of_incorp is a state in the United States, then the currency used for reporting "net income" and "total assets" will be "U.S. Dollars". If on the other hand, if it is some foreign country, then the currency will be the same as that which is in use in that country. For comparison purposes, portions of the context USER are also shown in Figure 8.

The representation of contexts as entities distinct from actual import/export schemas has several advantages. First, conflicts are dealt with at the level of semantic domains as opposed to attributes. Within a given import/export schema embedded in a single context, many attributes potentially map to the same semantic domain. Using this approach, the semantics of underlying data need be defined *at most* once[10]. This is the case with the context associated with Worldscope: instead of having to define scale-factors and currencies for "net income" and "total assets" (and possibly many others), these rules for interpreting monetary amounts need only be stated once. This stands in contrasts with shared-schema approaches in which the same conversion function has to be defined for each occurrence of an attribute from the same semantic domain (see for instance, the view definition for Approach 2 in Section 3).

Second, because the semantics are now being defined independently of specific schemas, it becomes feasible to document the assumptions and practices of an organization (or groups of organizations) in a manner that is independent of particular data sources. This organizational context can now be *inherited* and augmented by various specific departments within an organization. In adopting the organizational context as an outer context, a department inherits all the assertions of the organization and may optionally supplement it with additional constraints peculiar to the local situation. In this manner, an information infrastructure embodying a *hierarchy of contexts* can be realized.

Third, users are better empowered since they can query data sources directly (using the available export schemas) without necessarily committing to an import schema (or some predefined view on the data sources). Whenever data is exchanged between two contexts (as in doing a join between two sources, or when data is returned from a source to the receiver), the underlying semantic domains matching the data are compared to determine if there is any conflict. If this is the case, corrective actions can now be taken by a context mediator; in this manner, the users no longer need to be aware of the conflicts which may be present among data sources that are being accessed.

Finally, this representation scheme provides for better maintainability since relevant information about semantic domains are collected in one place. For instance, if the underlying organization changes its reporting practices such that all monetary amounts are to be reported in a different currency (say, changing from French francs to ECUs). This change need only be

[10] "at most once" because it can come for free, as when there is an outer context: see the next point.

made in one place (i.e., the organizational context) as opposed to every view definition which encapsulates the currency conversion function.

4.3 Schematic Mappings

In order for any of the knowledge described earlier to be useful, we need to know how an import/export schema is related to the underlying shared ontologies. The mappings between schemas and ontologies are done at two levels: (1) every attribute in an export/import schema is mapped to some semantic domain; and (2) every relation in an export/import schema is translated to a relational calculus expression using the vocabulary defined in the underlying ontologies. This is exemplified in Figure 9, which shows the schematic mappings for `Worldscope:AF` (the export schema for the Worldscope database of Figure 2) and an import schema, called `myView` (analogous to the one used in Section 3 but defined with reference to the ontology shown in Figure 6).

Worldscope:AF

`mapped_to(company_name,company_name).`
`mapped_to(location_of_incorp,location_name).`

$\vdots$

$\forall$ *Cname, Tic, Ctry, Ni, Ta*
`Worldscope:AF`(*Cname, Tic, Ctry, Ni, Ta*) $\Rightarrow$
$\exists$ *Comp, Lstg, Tr* `company_name`(*Comp, Cname*) $\wedge$ `net_income`(*Comp, Ni*)
$\wedge$ `total_assets`(*Comp, Ta*) $\wedge$ `listing`(*Comp, Lstg*) $\wedge$ `ticker`(*Lstg, Tic*).

myView

`mapped_to(company_name,company_name).`
`mapped_to(income,net_income).`

$\vdots$

$\forall$ *Comp, Cname, Ni, Ta, Lstg, Trd, LastP*
`company_name`(*Comp, Cname*) $\wedge$ `net_income`(*Comp, Ni*)
`total_assets`(*Comp, Ta*) $\wedge$ `listing`(*Comp, Lstg*)
$\wedge$ `trades`(*Lstg, Trd*) $\wedge$ `latest_trade_price`(*Trd, LastP*)
$\Rightarrow$ `myView`(*Cname, Ni, Ta, LastP*).

Figure 9: Schematic mappings for `Worldscope:AF` and `myView`.

Consider an earlier query in Section 3 in which `net_income` was to be retrieved from the `Worldscope` database and returned to the user. The `mapped_to` axioms allow us to first determine what the attribute `net_income` in the schema actually refers to[11]. Once the underlying semantic

[11]This appears less critical here because attribute names have been chosen to be self-explanatory to aid in the discussion. In many real systems, bewildering naming conventions abound and it is not uncommon to find

domain has been identified, the set of constraints placed on that concept (as asserted in the source and receiver contexts) can be used as the basis for determining whether or not a conflict exists. The details of the underlying mediation algorithm is the subject for the next section.

The relational calculus expression associated with each relation serves the purpose of translating between schemas and the vocabulary of the underlying ontologies. For example, a query formulated against the import schema `myView` can be reformulated as a calculus expression over the underlying shared ontologies, and subsequently rewritten as calculus expressions involving only predicate symbols corresponding to relation names in some underlying data sources. Similar kinds of translations may be needed in various other places: for instance, currency conversion uses conversion rates without naming what data source supplies them, and assignments to modifier values do so with reference only to some ontology and not specific data sources. We will describe examples of these and illustrate how the transformations can be achieved in Section 5.1.

5 Reasoning about Data Semantics

In the preceding section, we have described how different types of semantic knowledge can be represented within a Context Interchange system. The goal of these representations is to provide the knowledge needed for *context mediation*: i.e., the automatic detection and reconciliation of conflicts whenever data from disparate sources are combined and presented to receiver in a different context. A context mediator achieves this in four steps. If the query at hand does not state explicitly what data sources are used, a *resource discovery* step is used to identify the actual sources which are needed to evaluate the query. Following this, potential conflicts (and means for their reconciliation) have to be determined. We refer to this as *conflict detection* and this results in a *conflict table.* The next step is *query planning and reformulation* whereby the original query is reformulated into subqueries directed at individual data sources, and appropriate conversion operations (identified in the conflict table) can now be inserted in the appropriate places in the query tree. This step also encompasses optimization of the query: i.e., attempting to minimize the cost of executing the query (including the cost of data conversions). The final step is *execution* of the query plan which includes assembly of partial results and possibly conversion of the final answer to the context of the receiver.

The goal of this section is to illustrate how the representations chosen in the previous section can be used to facilitate context mediation. Specifically, we present an example which walks through all the above steps performed by a context mediator, then present in greater detail the actual algorithm used for conflict detection.

---

names like `RO-MLS-INC`.

5.1 Example

Suppose we have the following import schema:

```
myView(company_name,net_income,total_assets,last_trade)
```

which is a view defined over the "financial reporting" ontology. The schematic mappings for this view definition are shown in Figure 9 in Section 4.3. Notice that this definition does not name any specific data sources and references only the conceptual model presented in the ontology.

Consider the query

```
select company_name, net_income, last_trade
from myView
where total_assets > 1000000
```

assuming that the user issuing this query has a context named USER which assumes that (1) net_income and last_trade have scale_factor of 1; (2) total_assets has scale_factor 1000; and (3) all have currency 'British Pounds' as shown in Figure 8 (Section 4.2).

The first step in processing this query is to identify the actual data sources which can be used to obtain the answers. We refer to this as *resource discovery*. This is achieved by writing the query in clausal form:

$$\leftarrow \texttt{myView}(Cname, Ni, Ta, LastP) \wedge Ta > 1000000$$

and *unfolding* it using the schematic mapping for myView (Figure 9) to yield:

$$\leftarrow \texttt{company\_name}(Comp, Cname) \wedge \texttt{net\_income}(Comp, Ni) \wedge \texttt{total\_assets}(Comp, Ta) \wedge \texttt{listing}(Comp, Lstg) \wedge \texttt{trades}(Lstg, Trd) \wedge \texttt{latest\_trade\_price}(Trd, LastP) \wedge Ta > 1000000$$

and finally *folding* this clause using schematic mappings of data sources which are available. For example, using the mappings for Worldscope:AF and Networth:Stock yields the following:

$$\leftarrow \texttt{Worldscope:AF}(Cname, Tic1, Ctry, Ni, Ta) \wedge \texttt{Networth:stock}(Cname, Tic2, LastP) \wedge Ta > 1000000$$

This in turn can be translated to the SQL query

```
select w.company_name, w.net_income, n.lastTrade
from Worldscope:AF w, Networth:stock n
where w.company_name = n.companyName and w.total_assets > 1000000
```

The fold and unfold operations can be viewed as corresponding to resolution and inverse resolution [21]. In general, the resulting query is not unique since the same data may be obtained from multiple sources. In many instances, a user may want to add restrictions as to what data sources can be used to supply answers to a query. This can be accomplished by using schematic mappings from those databases as input to the fold/unfold transformation. This transformation can be augmented with cost information to identify the least-cost plan for a given query. The use of cost information in resource selection is being examined in our current research.

For the example at hand, step 1 of this mediation process has transformed a query formulated using an import schema to one which references the export schema of some underlying data sources. (With reference to Figure 3, this transforms a query under Approach 4 to an equivalent one formulated under Approach 3.) The goal now is to identify the potential conflicts which may occur if this query were to be executed without mediation, and then identify the means by which these conflicts can be mediated.

Since the user query is formulated in the `USER` context, the "normal" interpretation of the query is correct only if both `Worldscope:AF` and `Networth:Stock` are also in the `USER` context. Hence, a brute force approach to mediating the query is to first identify all relevant conflicts between the contexts for the sources (`WORLDSCOPE` and `NETWORTH`) and that of the receiver (`USER`), convert all sources to the same context, and then execute the query as if there is no conflict. This scheme is illustrated in Figure 10.

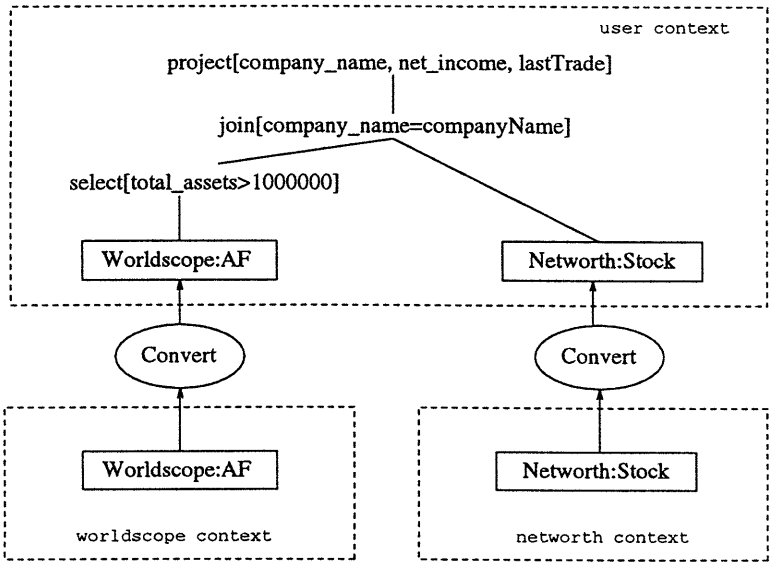

Figure 10: A conceptual view of query mediation.

In most cases, the brute force approach would be prohibitively expensive. Fortunately, we can do better using a number of optimization techniques. For example, selections and projections can be "pushed across" the conversion node to minimize the cost of conversion operations. The exact technique for query optimization we have adopted has been described elsewhere [8].

The identification of conflicts between two contexts is the main focus of the section. Conflict detection is accomplished using an *abductive* reasoning approach which we describe in detail in the next section. The conflict detection algorithm returns a conflict table which contains the details of what conflict exists and what conversion functions can be used to resolve these conflicts. The conflict table (with respect to the contexts `WORLDSCOPE` and `USER`) for the example at hand is shown in Figure 11. Each line of the conflict table identifies a conflict with respect to a given attribute/modifier which must be resolved. The "precondition" identifies the circumstances under which a conflict may occur; different preconditions may lead to different source and/or target modifier values and therefore may require different conversion functions. With reference to the sample conflict table in Figure 11, we know that total assets is in fact reported in `'U.S. Dollars'` when we are dealing with a U.S. company (i.e., when `location_of_incorp` is actually a U.S. state name), and reported in the country's currency if it is a foreign company. The boolean value reported under the heading "Invertible?" indicates whether or not the conversion function can be applied in reverse: i.e., getting from X' to X. This information is useful in determining if a constant selection constraint can be "pushed across" a conversion operation [8]. Based on this conflict table alone, we will be able to generate the subquery:

```
select w.company_name, w.net_income*1000*e.rate
from worldscope:AF w, almanac:US_States s, almanac:exchange e
where w.location_of_incorp=s.state and e.source='U.S. Dollar'
and e.target='British Pound'
and w.total_assets>1000000 / e.rate
union
select w.company_name, w.net_income*1000*e.rate
from worldscope:AF w, almanac:currencies c, almanac:exchange e
where w.location_of_incorp=c.country and e.source=c.currency
and e.target='British Pound'
and w.total_assets>1000000 / e.rate
```

This subquery then forms the basis for another join with `networth:stock` where another conflict involving `company_name` will be uncovered, causing the join to be made using the ticker symbol (i.e., attribute `ticker`) instead. The final query after taking into account this latter conflict will be identical to the one shown for Approach 1 in Section 3.

Attribute	Modifier	Src	Tgt	Precondition	Conversion Function	Invertible?
net_income	scale_factor	1000	1	(none)	$X' = X \times 1000$	yes
net_income	currency	'U.S. Dollar'	'British Pound'	w.location_of_incorp = s.state	e.source = <*Src*> and e.target = <*Tgt*> and X' = X*e.rate	yes
net_income	currency	<*variable*>	'British Pound'	w.location_of_incorp = c.country and c.currency = <*Src*>	e.source = <*Src*> and e.target = <*Tgt*> and X' = X*e.rate	yes
total_assets	currency	'U.S. Dollar'	'British Pound'	w.location_of_incorp = s.state	e.source = <*Src*> and e.target = <*Tgt*> and X' = X*e.rate	yes
total_assets	currency	<*variable*>	'British Pound'	w.location_of_incorp = c.country and c.currency = <*Src*>	e.source = <*Src*> and e.target = <*Tgt*> and X' = X*e.rate	yes

Figure 11: Conflict table between WORLDSCOPE and USER contexts for the example at hand. For brevity, the following aliases are used for relation names: w for Worldscope:AF, e for Almanac:Exchange, c for Almanac:Currencies and s for Almanac:US_States.

5.2 Conflict Detection As Abductive Reasoning

In this section, we present a conflict detection algorithm based on abduction. The term "abduction" refers to a particular kind of hypothetical reasoning which, in the simplest case, take the form:

> From observing A and the axiom $B \rightarrow A$
> Infer B as a possible "explanation" of A.

For instance, given the axioms (1) "it rained" $\Rightarrow$ "the floor is wet", and (2) "sprinkler was on" $\Rightarrow$ "the floor is wet", the observation of a "wet floor" will lead us to conclude that "it rained" or that "sprinkler was on". Although an explanation generated via an abduction step is usually unsound taken by itself, their collective disjunction is a sound inference if one assumes that the theory under consideration is closed [5]. For example, having observed that "the floor is wet", the conclusion ("it rained" $\vee$ "sprinkler was on"[12]) is perfectly logical if we make the assumption that there are no other reasons for a wet floor. Various forms of abductive reasoning have been applied to several AI problems (e.g., diagnostic reasoning, natural language understanding, and planning). A prominent application to databases has been reported in the area of database view updates [13].

Following Eshghi and Kowalski [9], we define an *abductive framework* to be a triple $<T, I, A>$ where T is theory, I is a set of integrity constraints, and A is a set of predicate symbols, called *abducible* predicates. Given an abductive framework $<T, I, A>$, an *abductive explanation* for Q is a statement E satisfying the following:

1. $\{E\} \cup T \Rightarrow Q$;
2. $\{E\} \cup T$ satisfies I; and
3. all predicates in E are elements in A.

For the purpose of conflict detection, we treat every ontology and context as a "theory" comprising of a collection of Horn clauses. Each "theory" is consistent taken by itself, but may be inconsistent with one another. (For example, `net_income` may be reported using a scale-factor of 1 in one context, and have a scale-factor of 1000 in another.) We use the notation

$$ist(t, p)$$

to mean that the clause p *is true* in the theory t[13]. For example, given that the clause

[12] The symbol $\vee$ refers to logical disjunction.

[13] (With apologies to McCarthy [19]): this notation is introduced by McCarthy in his work on "formalizing context". Unfortunately, he uses the word "context" in a slightly different way. Without getting entangled in the details, the notation remains a useful way for thinking about the relationship between different sets of statements which are internally consistent.

$$\texttt{scale\_factor}(X,1) \leftarrow \texttt{monetary\_quantity}(X).$$

is defined in the WORLDSCOPE context, we may then write

$$ist(\texttt{WORLDSCOPE}, \texttt{scale\_factor}(X,1) \leftarrow \texttt{monetary\_quantity}(X))$$

to signify that the statement is true in that "theory" but not necessarily so elsewhere. "Theories" can be nested: a theory t_1 is said to be nested in another t_2 if

$$\forall p:\ ist(t_2, p) \Rightarrow ist(t_1, p)$$

Hence, an ontology which references another is said to be nested in the latter by this definition. Similarly, "nested" contexts correspond to nested "theories"[14].

Before describing how the abductive framework can be applied to conflict detection, we need to first define precisely what a semantic conflict is. Let C_1 and C_2 be two contexts which references a concept P in an ontology G. Suppose M is a modifier of P. We say that a *conflict* exists between contexts C_1 and C_2 over P with respect to M if there exists some instance X such that

$$ist(G, P(X) \wedge ist(C_1, M(X,U)) \wedge ist(C_2, M(X,V)) \wedge U \neq V)$$

Intuitively, this says that the modifier M of P was assigned a value U in context C_1, and a *different* value V in C_2. For example, if net_income is reported in different currencies in Networth and Worldscope for at least some tuples, then it must be the case that there exists some instance X such that

$$ist(\texttt{FINANCIAL\_REPORTING}, \texttt{net\_income}(X) \wedge ist(\texttt{NETWORTH}, \texttt{currency}(X,U)) \\ \wedge\ ist(\texttt{WORLDSCOPE}, \texttt{currency}(X,V)) \wedge U \neq V)$$

Conflict detection can be construed as generating the abductive explanations (Es) for a hypothetical conflict Q. In other words, we assume that a conflict exists, then attempt to identify all scenarios that will lead to such a conflict. If no explanations for the conflict can be found, then there is simply no conflict. If on the other hand we end up with a nonempty set of explanations, each element in this set constitutes the (pre)conditions which, if found to be true, will lead to a conflict.

Figure 12 describes the conflict detection algorithm which is implemented for the current prototype. The input to the algorithm is the initial goal clause Q, and an abductive framework

[14] Given that we allow statements to be retracted, a more accurate characterization of "nesting" should be as follows: a "theory" t_1 is nested in t_2 if

$$\forall p:\ ist(t_2, p) \Rightarrow (\neg ist(t_1, \texttt{retract}(p)) \Rightarrow ist(t_1, p))$$

where '$\neg$' is to be interpreted as "negation as failure" [5].

$<T, I, A>$. The theory T corresponds to all statements of the form $ist(t, p)$ where p is a Horn clause and t is either a context or an ontology. The set I are the integrity constraints which are implicitly or explicitly present in the underlying conceptualizations[15]. A is a set of abducible predicates (which are really the relation names in the export schemas of available sources), with which we also associate the schematic mappings. The output from the algorithm consists of a set of explanations and the variable substitutions for U and V which are the values assigned to the modifiers in the respective contexts.

The execution of the algorithm can be understood in two parts. The first portion (the *while*-loop) finds an explanation for the hypothetical conflict, which the second transforms to a conjunctive expression using only abducible predicates (i.e., names of relations in the chosen export schemas) by folding the explanation with respect to the input schematic mappings. The algorithm terminates when all the choice points have been exhausted. Termination is guaranteed given a finite number of non-recursive[16] clauses in the theory.

The *while*-loop in the Conflict Detection Algorithm essentially implements a variant of *SLD-resolution* [16]. This differs from classical SLD-resolution in two ways. First, instead of failing at each branch whenever a literal cannot be further resolved, we cause the offending literal to be *abducted* (by adding it to *Exp*) if this does not lead to the violation of any integrity constraints. When an empty clause ($\square$) is finally derived, *Exp* contains an explanation for the goal we are attempting to prove. This implementation of abduction is similar to that taken by Cox and Pietrzykowski [7].

A second difference lies in the way in which a clause is chosen at the choice point (line 15). There are a number of subtleties here.

1. Not all the theories will be visible at the choice point. For example, to determine what scale-factors are used for monetary quantity in the `WORLDSCOPE` context, we must not end up using clauses which are asserted for a different context. We refer to the set of "theories" which are visible from a particular choice point as its *scope*. At any one time, this is determined by two things (1) the contents of the stack (*Stack*) which keeps track of what *ist*'s the current literal is being enclosed by, and (2) the nested structure of the "theories" themselves. To illustrate the preceding comments, suppose we have the goal

[15] A number of integrity constraints are implied by the way things are assigned to categories. For example, all modifiers have singleton values. Hence, if M is asserted to be a modifier for some semantic domain D, then whenever $M(X, U)$ and $M(X, V)$ hold, it must be that $U = V$ (i.e., there is a functional dependency here). Other constraints may be stated explicitly; e.g., we may have an integrity constraint (in the form of a denial)

$\leftarrow$ `location_of_incorp`($Comp$,$Loc1$) $\wedge$ `location_of_incorp`($Comp$,$Loc2$) $\wedge$ $Loc1 \neq Loc2$

which says that no companies can be incorporated in two different locations.

[16] This assumption is consistent with the fact that terminological cycles are disallowed in most knowledge representation systems.

Input Abductive Framework $<T, A, I>$, Goal Q of the form

$$ist(G, P(X) \wedge ist(C_1, M(X, U)) \wedge ist(C_2, M(X, V)) \wedge U \neq V)$$

(All variables are bound except for X, U and V.)

Output $S = \{<Exp, \theta>\}$ where

- θ is a substitution for variables U and V, and
- Exp is the explanation for the observed conflict $G\theta$.

Algorithm

```
1  Stack := φ; θ := {U/U, V/V}; Cur := Q; Exp := □;
2  while Cur ≠ □
3        do
4        assign the first literal to Lit and remove it from Cur;
5        case 1: Lit = ist(C, Stmt)
6                push(C, Stack);
7                Cur := Stmt∧ ↑ ∧Cur;
8        case 2: Lit =↑
9                pop(Stack);
10       case 3: Lit = expression involving a builtin operator
11               if evaluate(Lit) = false
12                 backtrack(); fi
13       case 4: Lit is an atomic predicate
14               /* nondeterministic choice point */
15               find a clause Head :- Body such that Head unifies with
16                                Lit using mgu ω
17               if exists
18                 then Cur := (Body ∧ Cur)ω;
19                      θ := θω;
20                  else if Lit can be added to Exp without
21                          violating any integrity constraints
22                          /* abduct Lit */
23                          Exp := Exp ∧ Lit;
24                     else backtrack()
25                          fi
26                 fi
28       od;
29 fold Exp by replacing all predicates with abducible ones;
30 print Exp, ω;
31 backtrack();
```

Figure 12: The Conflict Detection Algorithm

$ist(o_1, \ldots, ist(c_1, \ldots, L, \ldots))$

where L is some literal. At the choice point for L, *Stack* will have the values $<o_1, c_1>$. Suppose o_1 references some other ontologies o_2, o_3 and o_2 in turn references o_4. Furthermore, c_1 is a context nested in c_2, which references ontologies o_1, o_5, and no other contexts or ontologies are referenced directly or indirectly. In this case, the scope of the choice point is simply the set of all the "theories" above (o_1 to o_5, c_1 and c_2).

2. Once the scope of a choice point has been determined, any clause which is present in any one of the theories in the scope is a potential resolvent for the current literal L, except if that clause has been retracted. For example, context c_2 may assert that the `scale_factor` for `monetary_quantity` is '1', but this may be retracted in c_1. In this case, the Horn clause which is present in c_2 will not qualify (even though it is physically present).

As a first illustration of how the algorithm works, consider how conflicts concerning `scale_factor` of `net_income` might be uncovered. In this case, the query posed will be:

$$ist(\texttt{FINANCIAL\_REPORTING}, \texttt{net\_income}(X) \wedge ist(\texttt{WORLDSCOPE}, \texttt{scale\_factor}(X, U)) \wedge ist(\texttt{USER}, \texttt{scale\_factor}(X, V)) \wedge U \neq V)$$

Figure 13 shows the corresponding SLD-tree for this query. Note that the algorithm returns only one explanation:

$$Exp = \texttt{net\_income}(X)$$

which suggests that the conflict will always be true for any data element drawn from the `net_income` semantic domain. The remaining two branches which were explored fail to lead to a successful termination (an empty clause) since both attempt to abduct a literal which contradicts what has already been asserted (specifically, X cannot be asserted to be instances from two distinct domains, `net_income` and `total_assets`, simultaneously).

6 Conclusion

A variety of different issues have been addressed in this paper, all of which represent new results in the area of semantic interoperability using Context Interchange beyond those in previous papers [8,10,23,25]:

1. We have described the architecture of a Context Interchange Prototype which enables integrated access to data from a variety of sources on the World Wide Web. This prototype not only provides a concrete demonstration of our integration strategy, but also demonstrates

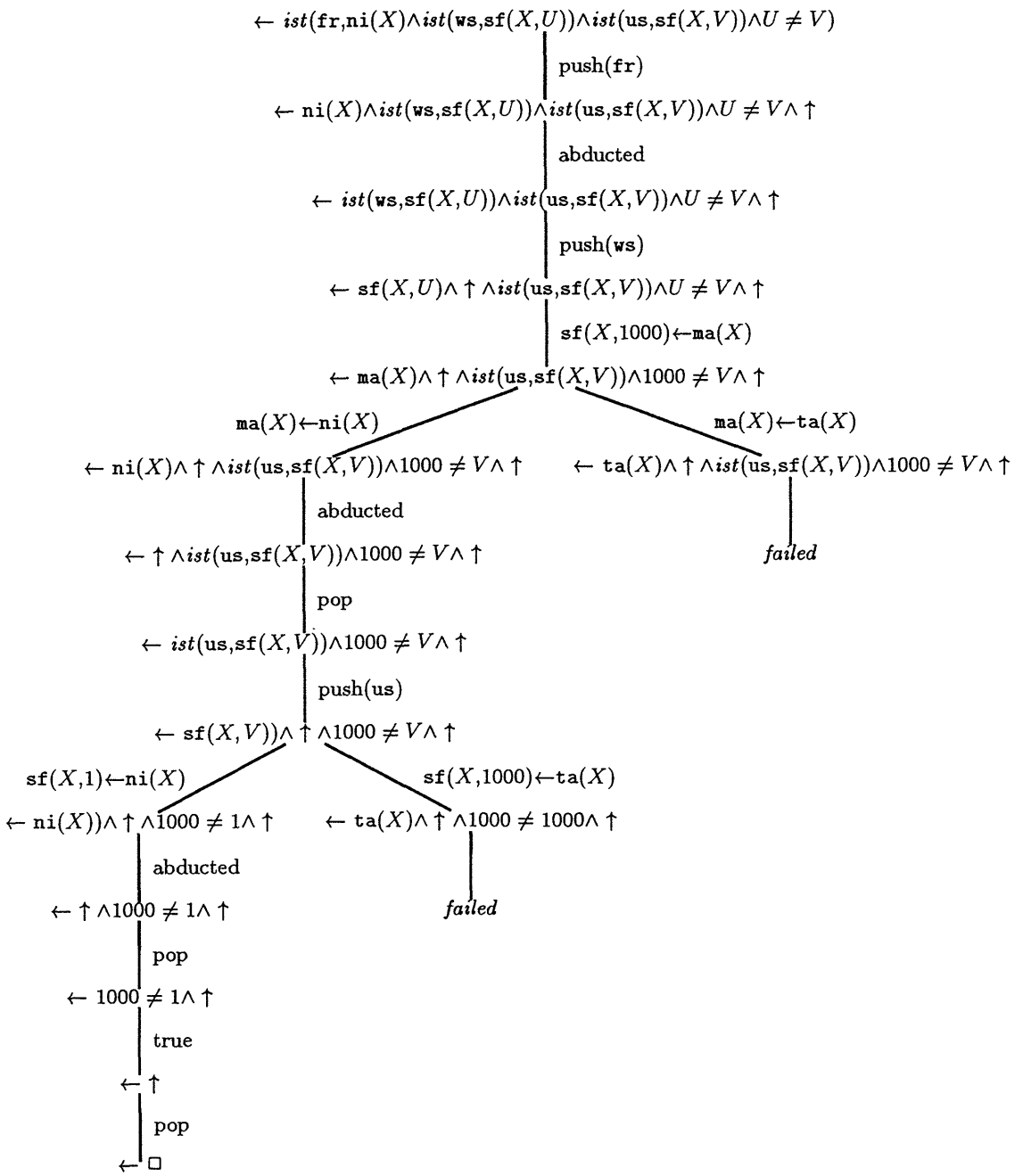

Figure 13: Determining conflict for `scale_factor` of `net_income` in two different contexts: `WORLDSCOPE` and `USER`. (Symbols are abbreviated for formatting reasons.)

the feasibility of exploiting an existing information infrastructure in providing mediation services on a large-scale.

2. Many in the integration community have held the view that automatic reconciliation of semantic conflicts can only be achieved if queries are formulated against some shared schema which reconcile the conflicts, a priori, in its (view) definition. We have argued that it is plausible, and sometimes desirable, for users to formulate a query based on export schemas provided by data sources while still enjoying the benefits of having the query mediated by the system. In other words, the presence of a import schema (or user view) is orthogonal to whether or not a query can be mediated.

3. We have proposed a three-tiered representation of semantic knowledge in which (1) *ontologies* provide descriptions of well-delimited domains; (2) *contexts* augment ontologies by identifying idiosyncratic assumptions or constraints pertaining to the local environment; and (3) *schematic mappings* provide the means of translating from one set of vocabulary (of database schemas) to another (that of the underlying ontologies). We claimed (and have illustrated with examples) that this representation makes it easy for semantic knowledge to be shared and reused for different purposes.

4. We have proposed a conflict detection algorithm based on the abduction framework which provides for the automatic detection of semantic conflicts between two different contexts. This is accomplished by treating ontologies and contexts as logical theories from which explanations to a hypothetical conflict is sought. By adopting a logical interpretation of knowledge present in ontologies, contexts and semantic mappings, we have a sound and parsimonious framework for inferring about potential semantic conflicts.

To realize the full potential of Context Interchange, a number of important issues need to be further researched. The field of "ontology engineering" remains immature and presents many challenging research issues. Many research problems remain for designing, building, and maintaining a set of shared ontologies. Other research issues are more oriented towards the database community. For instance, query planning and optimization in the context of heterogeneous database systems remains poorly understood. The heterogeneity and autonomy of the underlying data sources present a number of important issues which are not addressed by past research in distributed databases [17]. For instance, the non-availability of database statistics effectively renders many optimization algorithms useless. Similarly, it is difficult to acquire costs associated with data conversions (in circumventing the conflicts); when these are significant, it is important to avoid brute force transformation of large data sets.

The feasibility of the integration approach described in this paper has been demonstrated in our current prototype implementation. The large number of data sources on the World Wide Web, and their inherent heteorgeneity, will allow us to test our claims concerning the scalability of our approach. In the near future, this prototype will be used as a testbed for experimenting with solutions to a number of research problems, some of which have been outlined above.

References

[1] AHMED, R., SMEDT, P. D., DU, W., KENT, W., KETABCHI, M. A., LITWIN, W. A., RAFFI, A., AND SHAN, M.-C. The Pegasus heterogeneous multidatabase system. *IEEE Computer 24*, 12 (1991), 19–27.

[2] ARENS, Y., AND KNOBLOCK, C. A. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 92–101.

[3] BORGIDA, A. On the relationship between description logic and predicate logic queries. In *Proc of the Third International Conference on Information and Knowledge Management* (Gaithersburg, MD, Nov 29 – Dec 1 1994), pp. 219–225.

[4] BORGIDA, A., BRACHMAN, R. J., MCGUINNESS, D. L., AND RESNICK, L. A. Classic: a structural data model for objects. In *Proceedings of the ACM SIGMOD Conference* (Portland, OR, 1989), pp. 59–68.

[5] CLARK, K. Negation as failure. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, 1978, pp. 292–322.

[6] COLLET, C., HUHNS, M. N., AND SHEN, W.-M. Resource integration using a large knowledge base in Carnot. *IEEE Computer 24*, 12 (Dec 1991), 55–63.

[7] COX, P. T., AND PIETRZYKOWSKI, T. Causes for events: their computation and applications. In *Proceedings CADE-86* (1986), J. H. Siekmann, Ed., Springer-Verlag, pp. 608–621.

[8] DARUWALA, A., GOH, C. H., HOFMEISTER, S., HUSSEIN, K., MADNICK, S., AND SIEGEL, M. The context interchange network prototype. In *Proc of the IFIP WG2.6 Sixth Working Conference on Database Semantics (DS-6)* (Atlanta, GA, May 10 – Jun 2 1995). Forthcoming.

[9] ESHGHI, K., AND KOWALSKI, R. A. Abduction compared with negation by failure. In *Proc. 6th International Conference on Logic Programming* (Lisbon, 1989), pp. 234–255.

[10] GOH, C. H., MADNICK, S. E., AND SIEGEL, M. D. Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment. In *Proceedings of the Third International Conference on Information and Knowledge Management* (Gaithersburg, MD, Nov 29–Dec 1 1994), pp. 337–346.

[11] GRUBER, T. R. Ontolingua: a mechanism to support portable ontologies. Tech. Rep. KSL 91-66, Knowledge Systems Laboratory, Stanford University, 1992.

[12] GRUBER, T. R., AND OLSEN, G. R. An ontology for engineering mathematics. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning* (Gustav Stresemann Institut, Bonn, Germany, 1994), J. Doyle, P. Torasso, and E. Sandewall, Eds., Morgan Kaufmann. (These and other ontologies are accessible from `http://www-ksl.stanford.edu/knowledge-sharing/ontologies/index.html`.).

[13] KAKAS, A. C., AND MANCARELLA, P. Database updates through abduction. In *Proceedings 16th International Conference on Very Large Databases* (Brisbane, Australia, 1990).

[14] LANDERS, T., AND ROSENBERG, R. An overview of Multibase. In *Proceedings 2nd International Symposium for Distributed Databases* (1982), pp. 153–183.

[15] LITWIN, W., AND ABDELLATIF, A. An overview of the multi-database manipulation language MDSL. *Proceedings of the IEEE 75*, 5 (1987), 621–632.

[16] LLOYD, J. W. *Foundations of logic programming*, 2nd, extended ed. Springer-Verlag, 1987.

[17] LU, H., OOI, B.-C., AND GOH, C. H. On global multidatabase query optimization. *ACM SIGMOD Record 20*, 4 (1992), 6–11.

[18] MACGREGOR, R., AND BATES, R. The Loom knowledge representation language. In *Proceedings of the Knowledge-Based Systems Workshop* (1987).

[19] MCCARTHY, J. Notes on formalizing context. In *Proceedings 13th IJCAI* (1993).

[20] PATIL, R. S., FIKES, R. E., PATEL-SCHNEIDER, P. F., MCKAY, D., FININ, T., GRUBER, T., AND NECHES, R. The DARPA Knowledge Sharing Effort: progress report. In *Proceedings of the Annual International Conference on Knowledge Acquisition* (Cambridge, MA, October 1992). (The KSE public library is accessible at URL `http://www-ksl.stanford.edu/knowledge-sharing/README.html`.).

[21] PETTOROSSI, A., AND PROIETTI, M. Transformation of logic programs: foundations and techniques. *Journal of Logic Programming 19* (1994), 261–320.

[22] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. Context interchange using meta-attributes. In *Proceedings of the 1st International Conference on Information and Knowledge Management* (1992), pp. 377–386.

[23] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems 19*, 2 (June 1994), 254–290.

[24] SIEGEL, M., AND MADNICK, S. Context interchange: sharing the meaning of data. *ACM SIGMOD Record 20*, 4 (1991), 77–78.

[25] SIEGEL, M., AND MADNICK, S. A metadata approach to solving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases* (1991), pp. 133–145.

[26] SMITH, J., AND SMITH, D. Database abstractions: aggregation and generalization. *ACM Transactions on Database Systems 2*, 2 (1977), 105–133.

[27] WANG, R. Y., AND MADNICK, S. E. A polygen model for heterogeneous database system: the source tagging perspective. In *Proceedings of the 16th Conference on Very Large Data Bases* (1990), pp. 519–538.